Dynamo: Amazon's Highly Available Key-value Store

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, Werner Vogels

SOSP(2007)

Presenter: Shichao Jin

Outline

- Background
- Design Principles
- Techniques
- Conclusion

Background

- Amazon Shopping Carts
- low-latency key-value storage
  - Put() & Get()
  - SLA: response within 300ms for 99.9% of requests
  - hundreds of nodes
- a collection of distributed techniques
- spawned many imitators
  - Voldemort (LinkedIn)
  - Cassandra (Facebook)

Design Principles

- Always-writable
- Incrementally scalable
- Symmetrical
- Decentralized
- Heterogenous

Techniques

Problem	Technique
Partitioning	Consistent hashing
High availability for writes	Eventual consistency, Vector clocks with reconciliation during reads
Handling temporary failures	Sloppy quorum protocol and hinted handoff
Recovering from permanent failures	Anti-entropy using Merkle trees
Membership and failure detection	Gossip-based membership protocol and failure detection

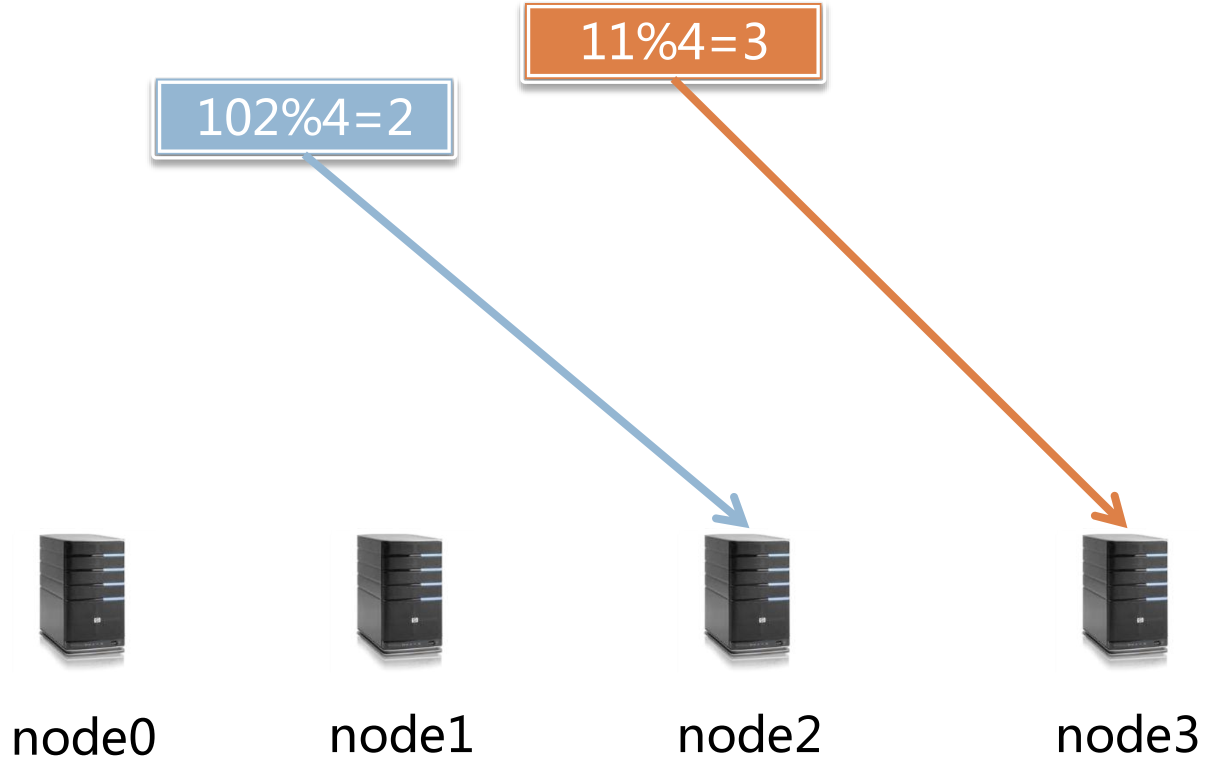
Partition——Consistent Hashing

- m nodes
- items identified by keys
- How to partition items to m nodes?

$$\text{hash}(\text{key}) \bmod m$$

Partition——Consistent Hashing

hash(key) mod m



Partition——Consistent Hashing

Disadvantages of hash:

- static, rehash when add/delete node(s)

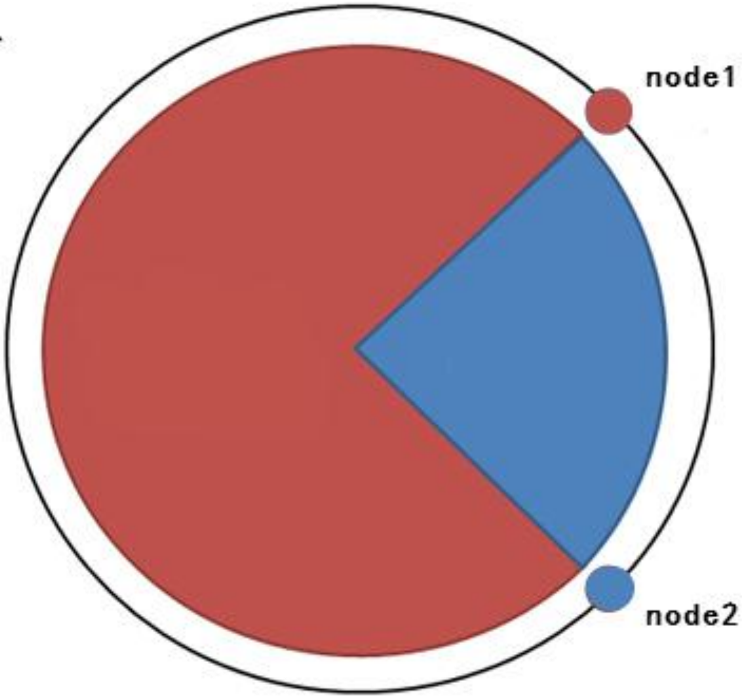
Solution:

- **Consistent Hashing**

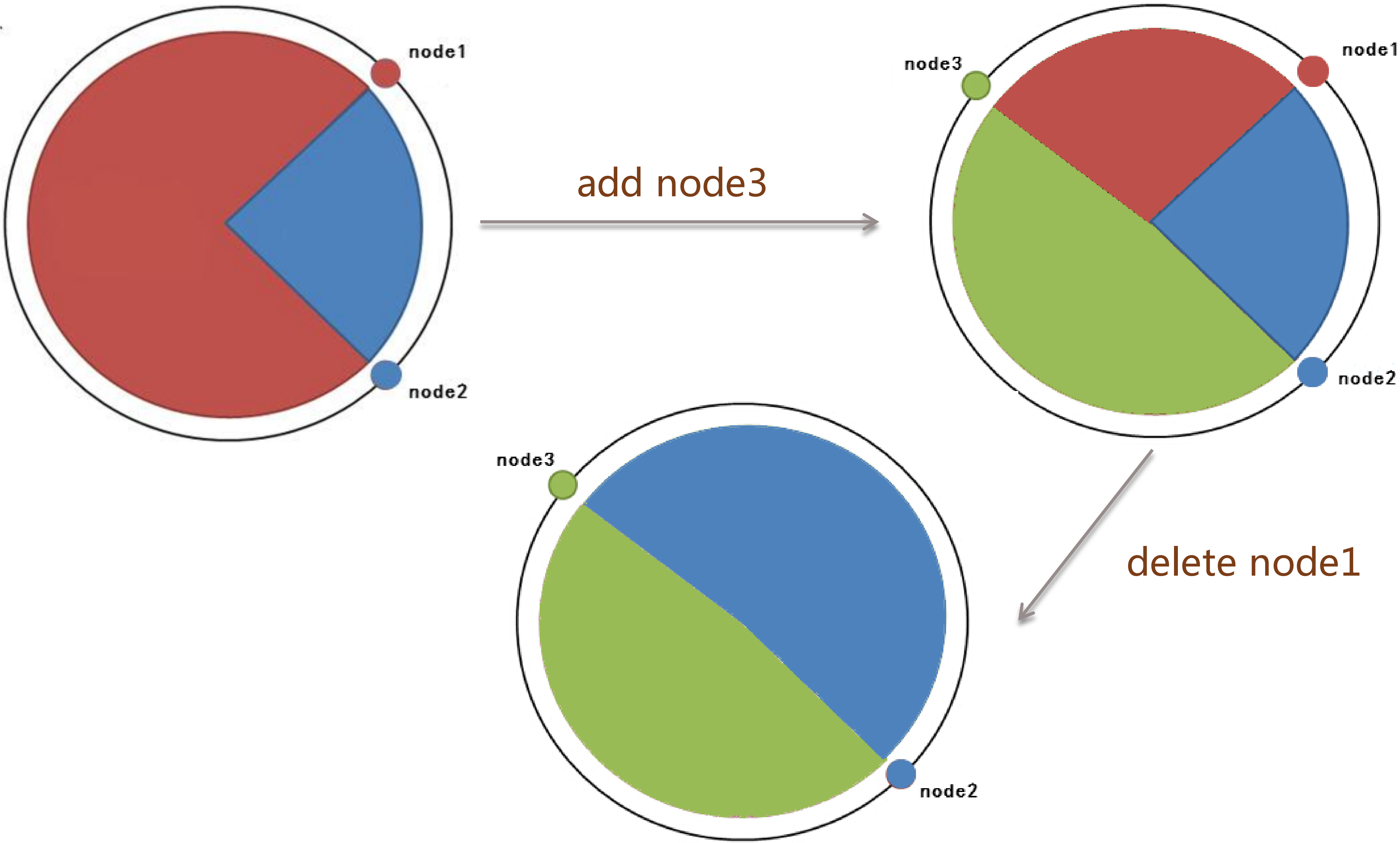
Partition——Consistent Hashing

Consistent Hashing:

- hash space: ring
- each node manages a region
- all rehash is unnecessary



Partition——Consistent Hashing



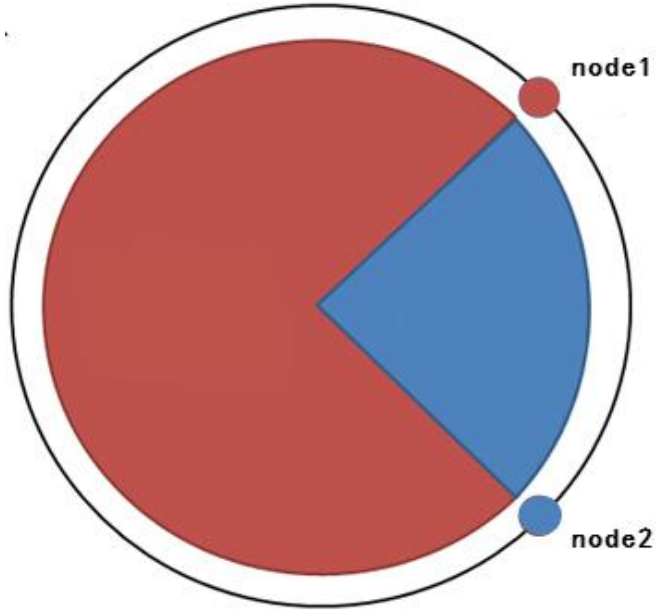
Partition——Consistent Hashing

Problems of Consistent Hashing:

- non-uniform load distribution
- heterogeneity

Solution:

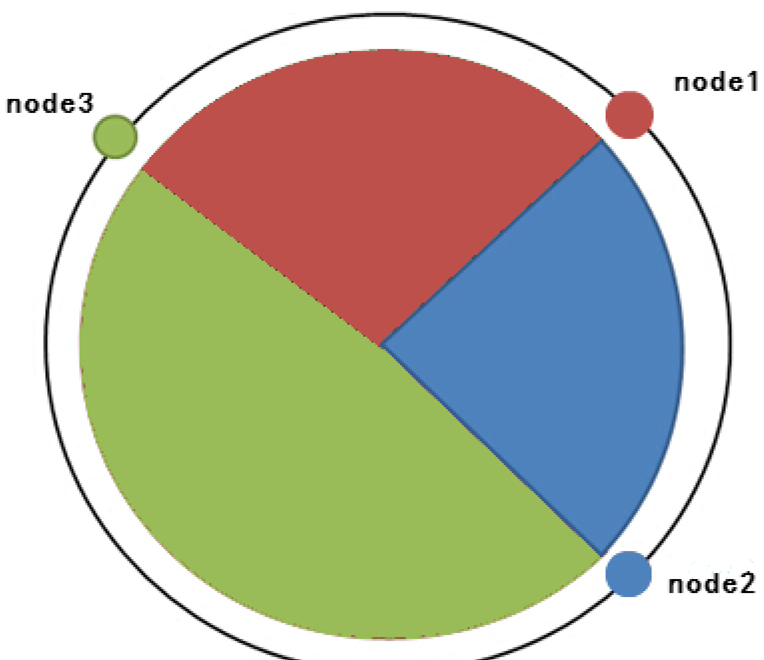
- **Virtual Nodes**

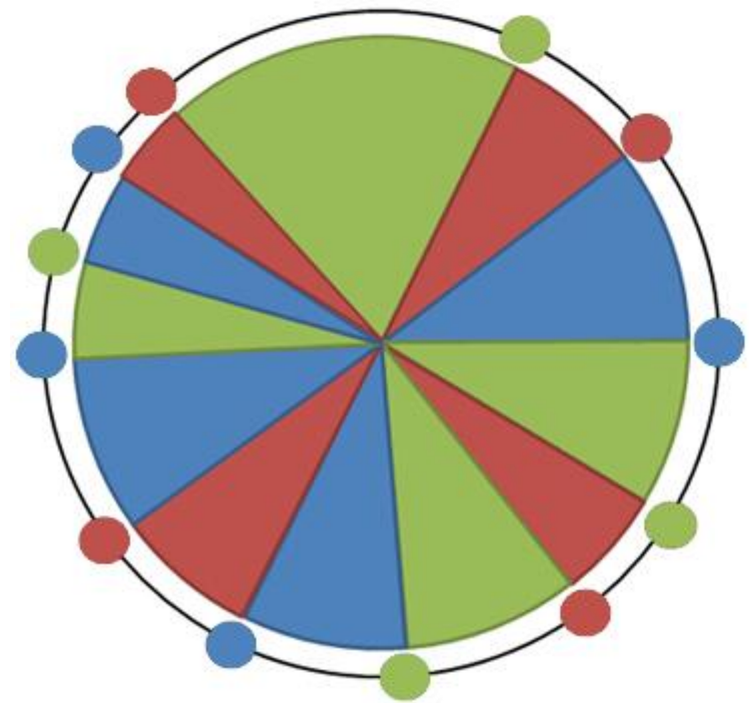


Partition——Consistent Hashing

Virtual Nodes:

- disperse load to other nodes when a node fails

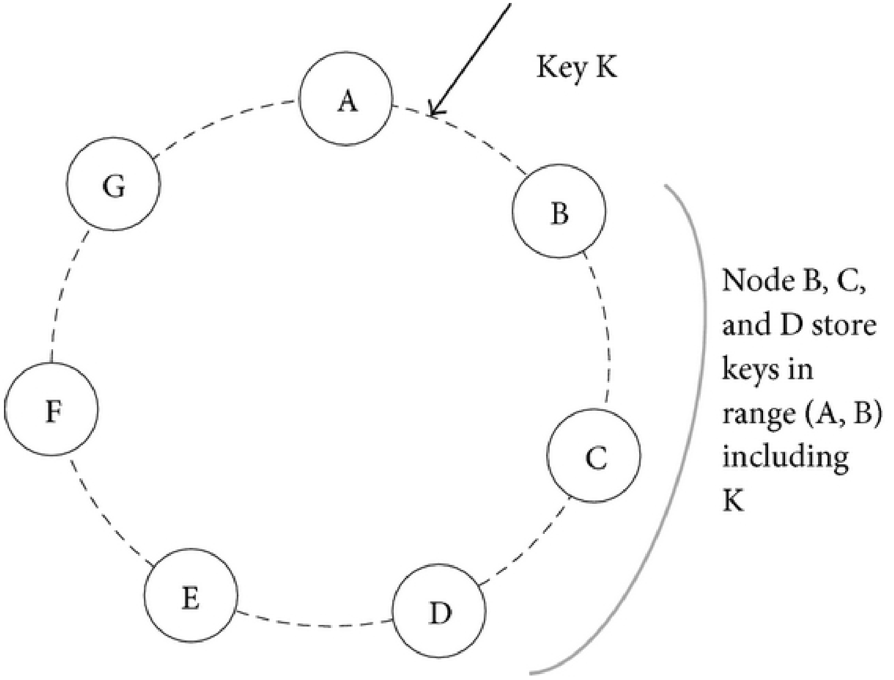




Replication

An Example for Replication

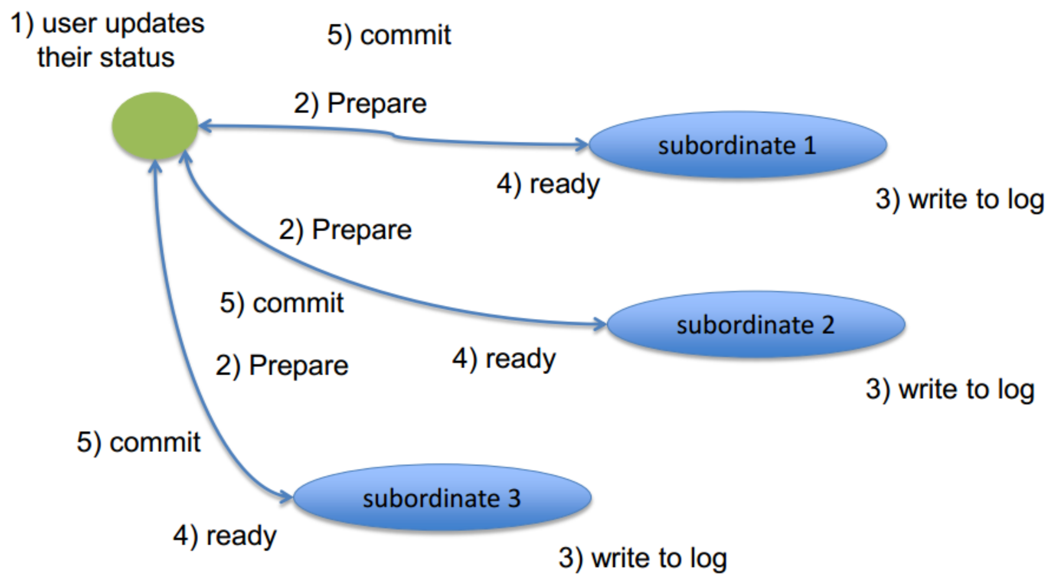
- N = 3
- B, C, D is K's preference list
- for fault-tolerance
- for availability



High Availability for Writes

Concurrent Writes:

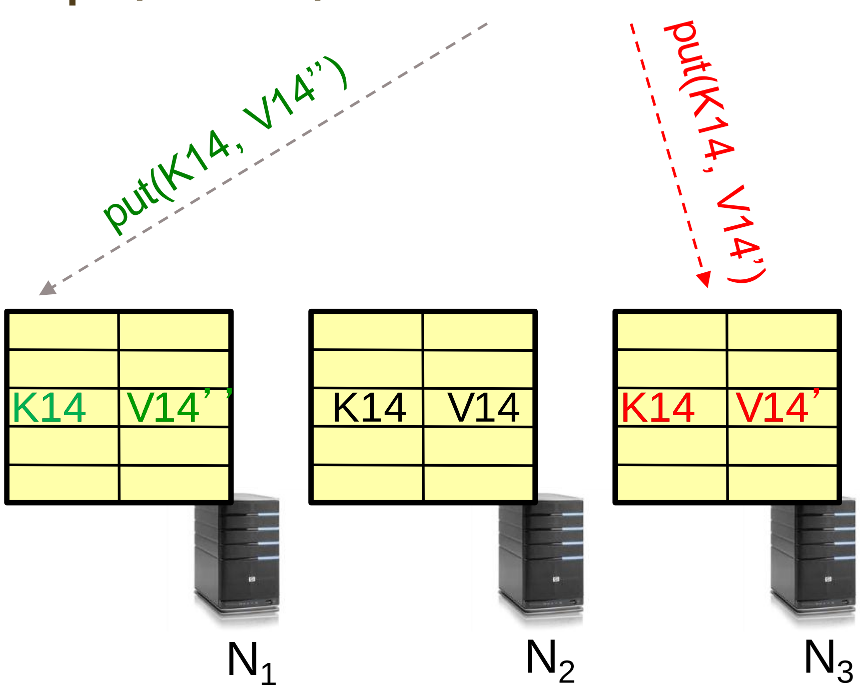
- Application: Shopping Cart
- Two-Phase Commit in distributed RDBMS



High Availability for Writes

Concurrent Writes:

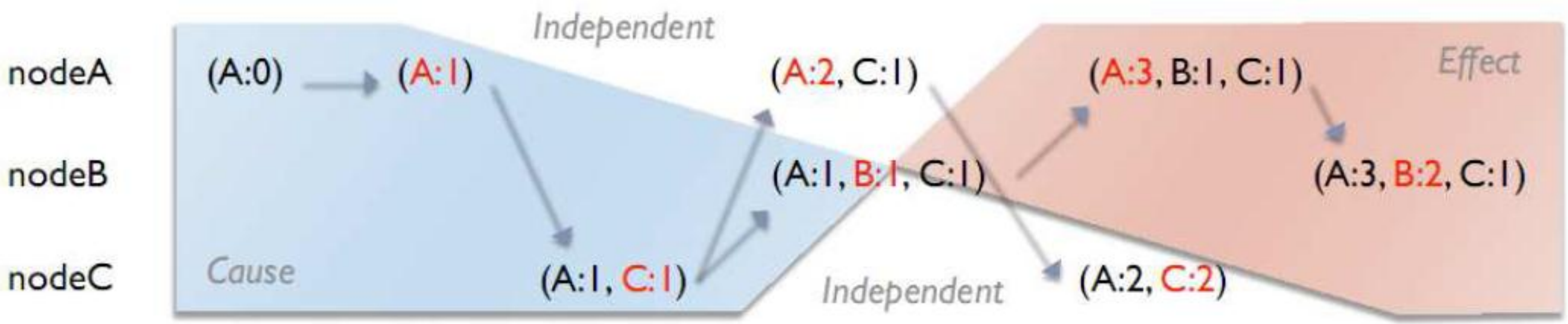
- Problem: 2 (more) versions of a data item
- Possible Solution: timestamp (How?)
- Dynamo: Vector Clocks



High Availability for Writes

Vector Clocks:

- logical clock
- causal order (partial)

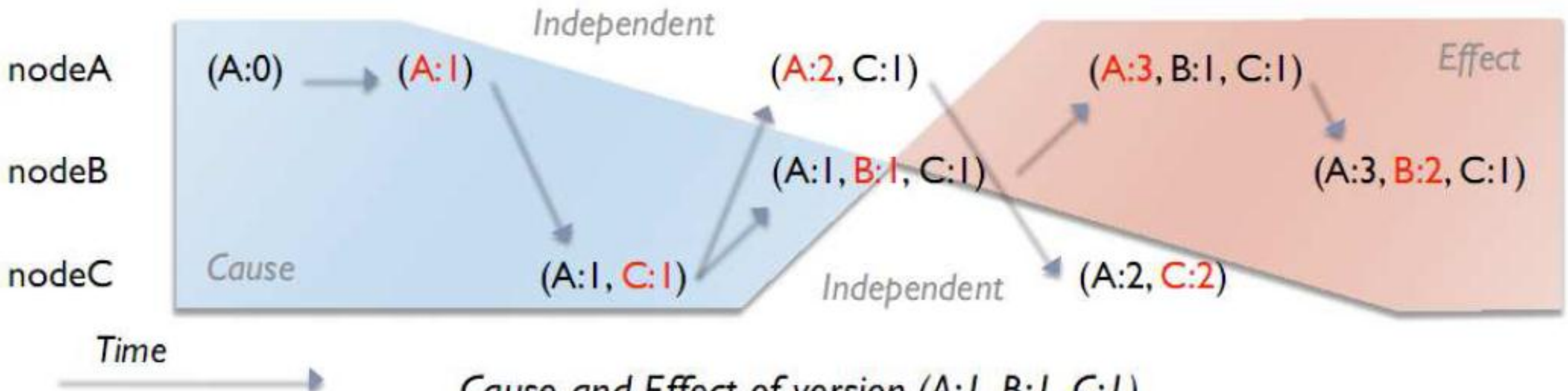


Cause and Effect of version (A:1, B:1, C:1)

High Availability for Writes

How to determine ordering of versions?

- (A:1, B:1, C:1) < (A:3, B:1, C:1)
- (A:1, B:1, C:1) ? (A:2, C:1)

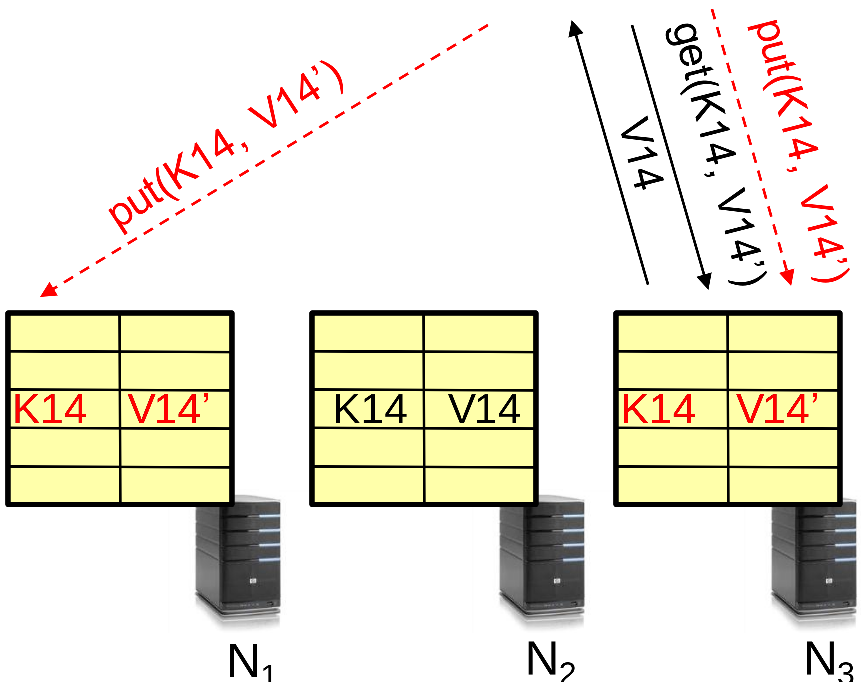


Cause and Effect of version (A:1, B:1, C:1)

Consistency——Strict Quorum

Eventual Consistency:

- given enough time all updates will propagate through the system
- Read after Write



Consistency——Strict Quorum

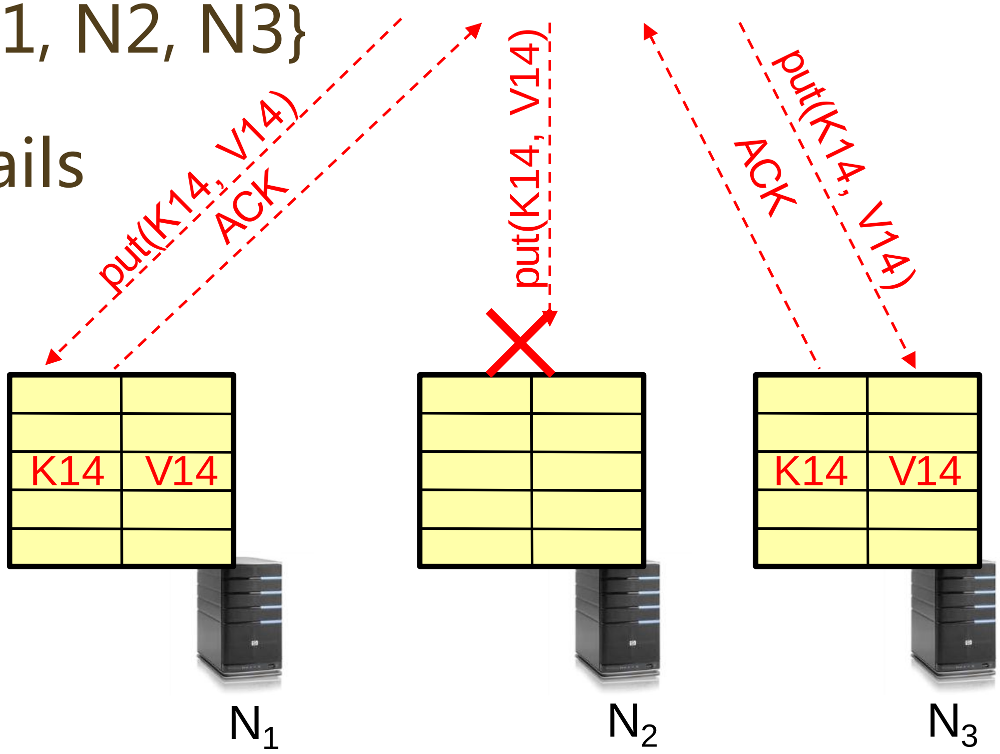
Strict Quorum:

- see the latest data
- define a replica set of size N
- put() waits for acks from at least W replicas
- get() waits for responses from at least R replicas
- $W+R > N$

Consistency——Strict Quorum

Strict Quorum Example:

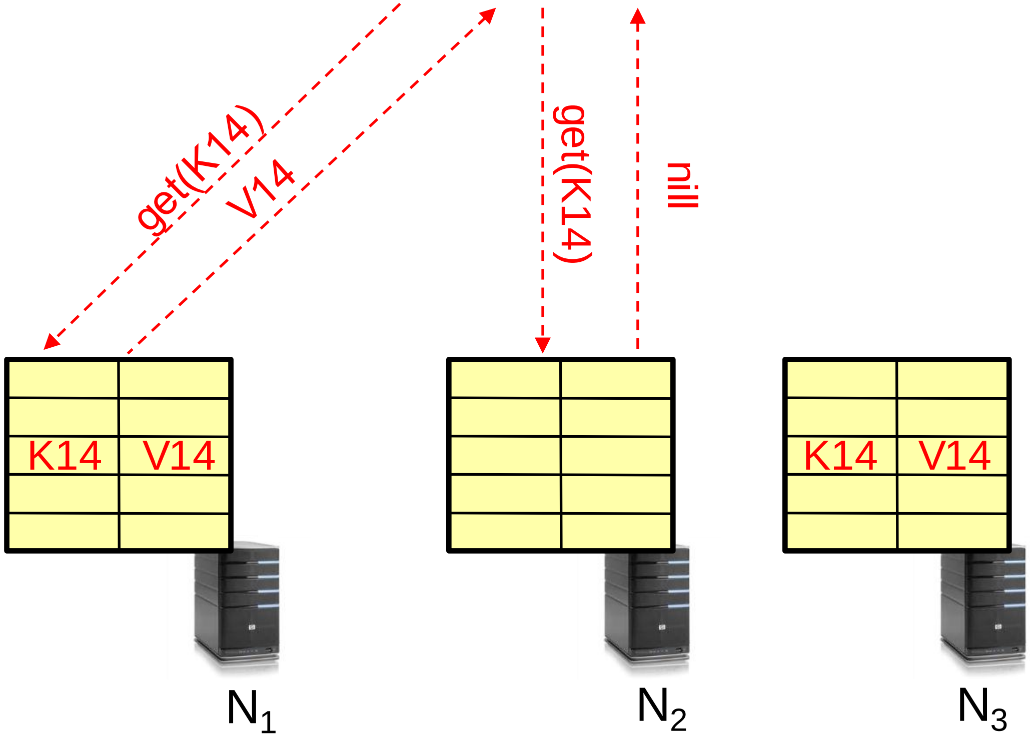
- N=3, W=2, R=2
- replica set for K14: {N1, N2, N3}
- assume put() on N3 fails



Consistency——Strict Quorum

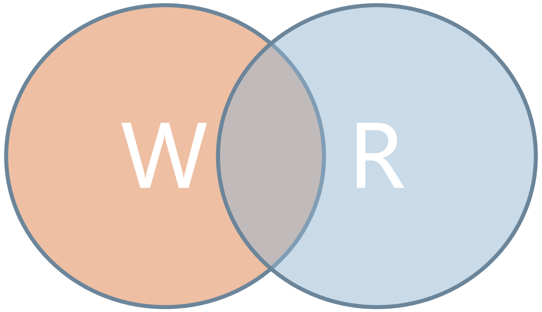
Strict Quorum Example:

- Now, issuing get() to any two nodes out of three will return the answer



Consistency——Strict Quorum

Why does Strict Quorum works?



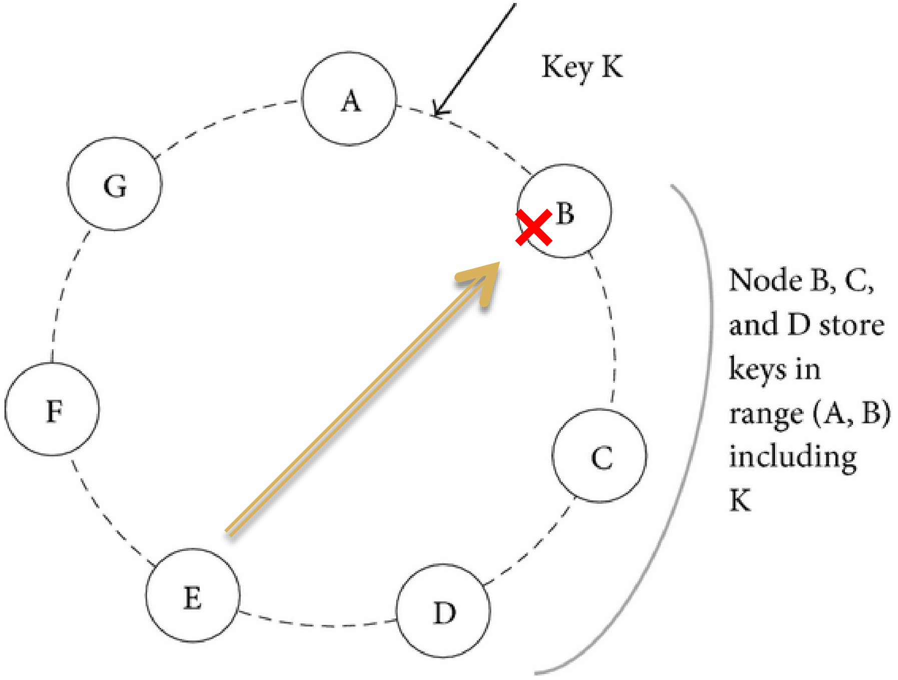
Tune W, R, N:

- optimized for write, set W small
- optimized for read, set R small

Temporary Failure ——Hinted Handoff

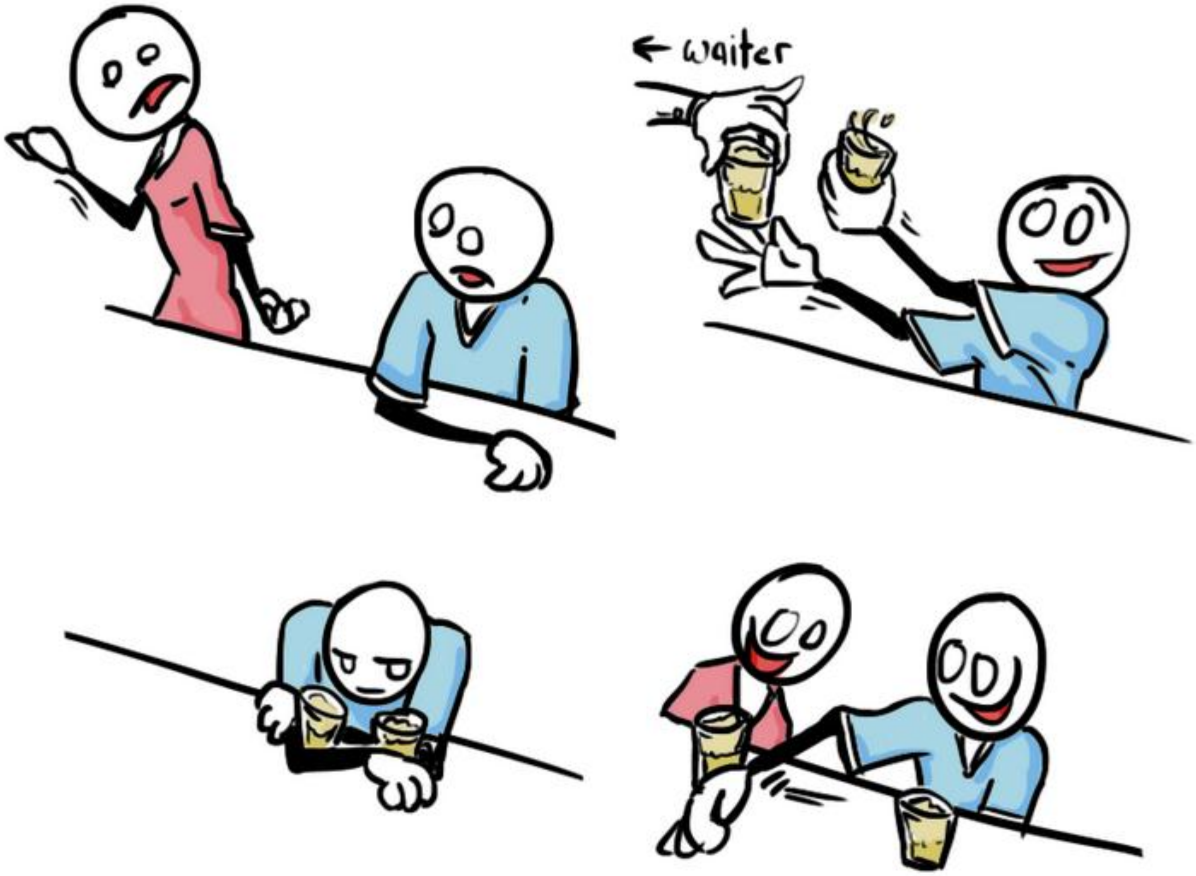
Hinted Handoff (Sloppy Quorum)

- node accepts writes for other down nodes
- data accepted by other node is handed off when down node recovers
- set W = 3, N = 3
- do not wait B recover



Temporary Failure ——Hinted Handoff

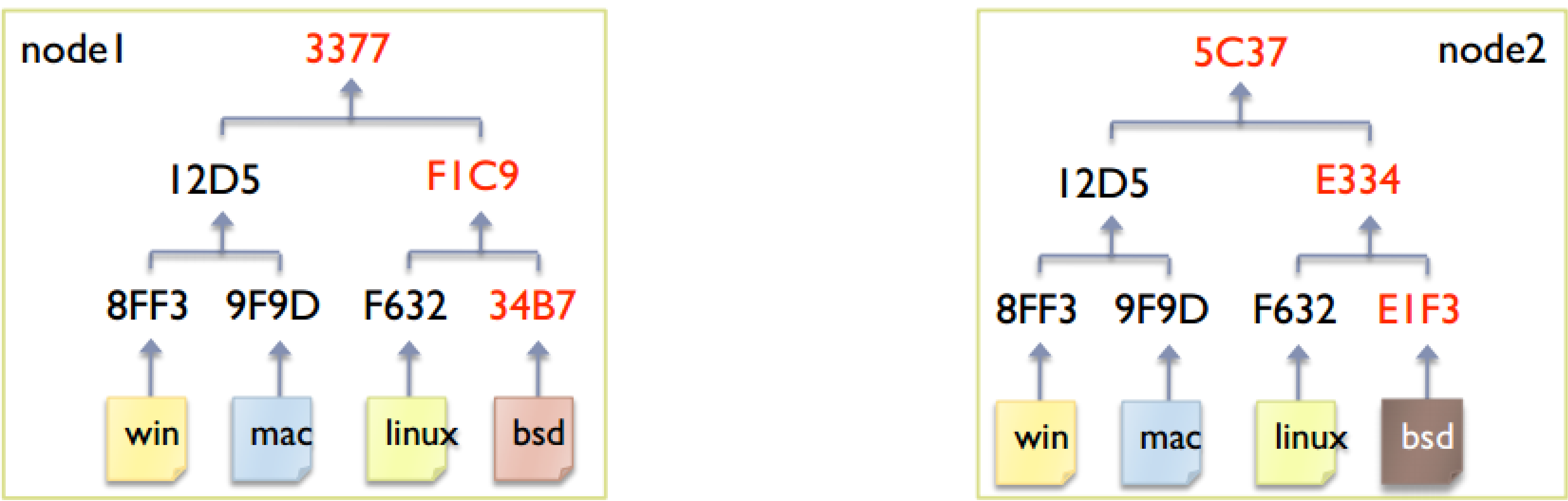
- Sloppy Quorum



Permanent Failure ——Replica Synchronize

Replica Synchronization (Merkle tree)

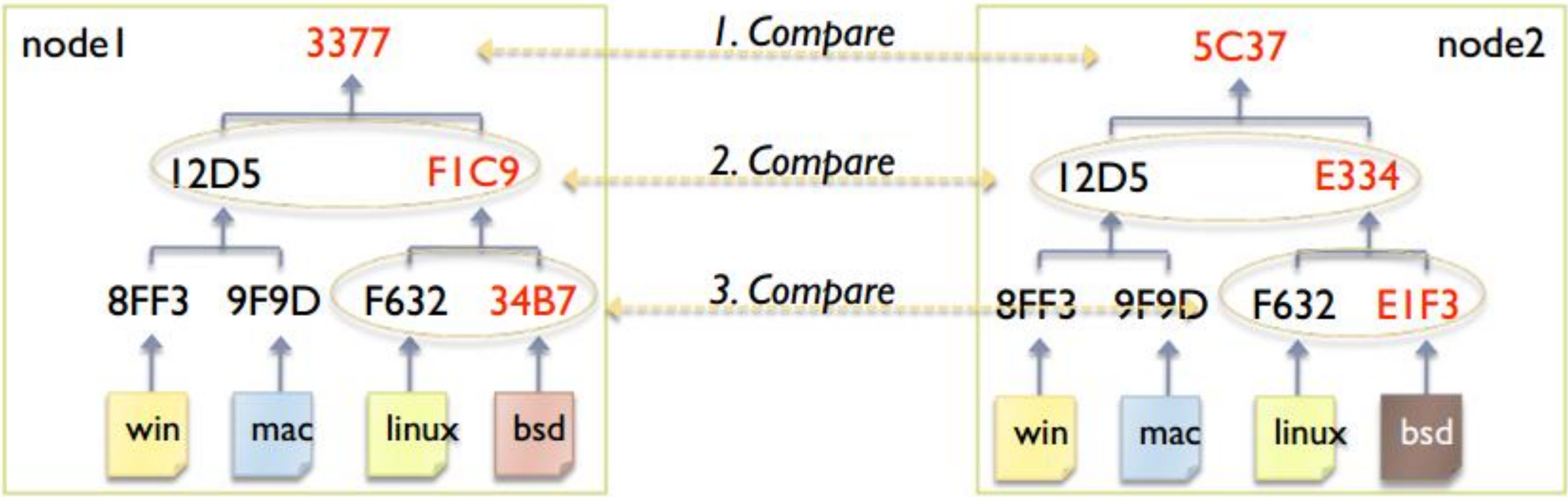
- hierarchical checksums
- executed periodically or when membership changes



Permanent Failure ——Replica Synchronize

Replica Synchronization (Merkle tree)

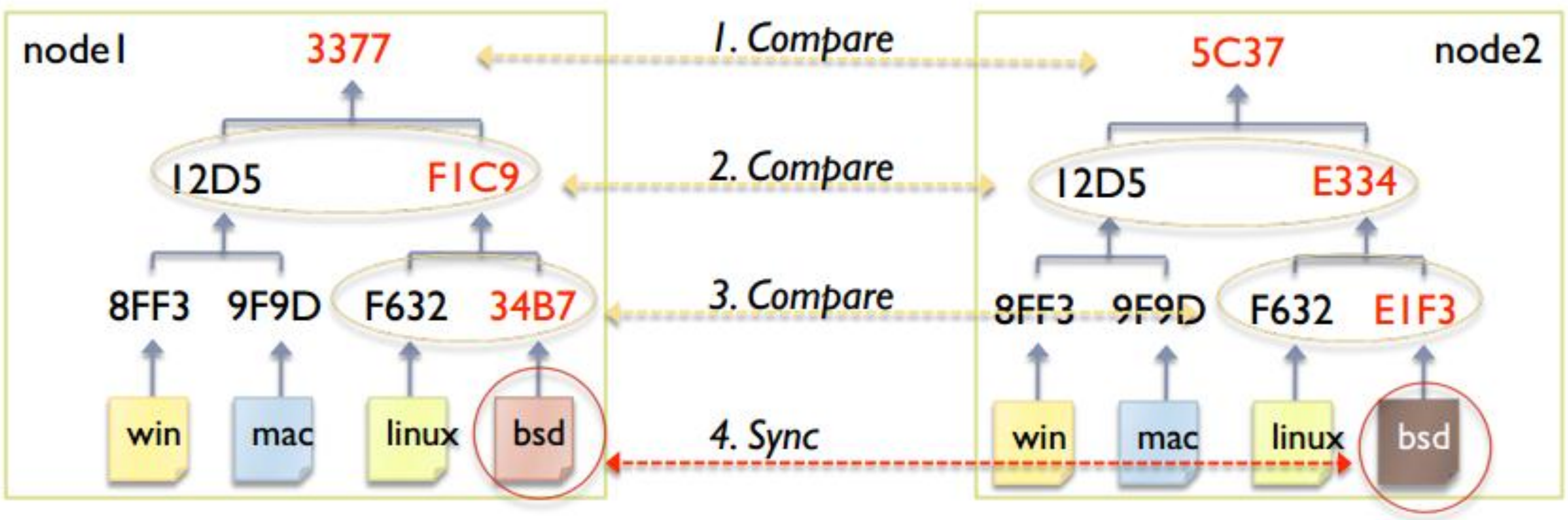
- hierarchical checksums
- executed periodically or when membership changes



Permanent Failure ——Replica Synchronize

Replica Synchronization (Merkle tree)

- hierarchical checksums
- executed periodically or when membership changes



Conclusion

- Consistent Hashing
- Vector Clocks
- Eventual consistency
- Strict & Sloppy Quorum
- Merkel Tree

References

- Dynamo Paper
- KaiAn: Open Source Implementation of Amazon’s Dynamo
- UCB CS162: Key-Value Store, Networking, Protocols
- A Little Riak Book by Eric Redmond

Q & A